1.Lanelet2 Paper Overview

1. Introduction to Lanelet2 Framework:

The paper introduces Lanelet2, an open-source map framework implemented in C++ designed to meet the high demands of maps for highly automated driving. It highlights the importance of high-definition maps in automated driving projects due to their ability to compensate for sensor limitations, provide information about non-observable regions, and ensure a reliable interpretation of sensor data.

The paper distinguishes between "lane-level accurate maps" suitable for navigation and partially automated driving on highways and less complex roads, and "highly automated driving (HAD) maps" that are needed for safe operation in complex urban environments.

In HAD maps, detailed information about the road environment, including bicycle lanes, sidewalks, markings, traffic signs, and curbs, is essential. Different components of an automated vehicle's processing chain rely on precise map data, such as localization, behavior generation, prediction of other road users, and detailed routing. The map must also handle special situations like emergency or parking maneuvers and support simulation under realistic conditions.

The paper emphasizes the need for high-quality, accurate, up-to-date, and verifiable maps, down to centimeter-level precision, to ensure safe automated driving. It points out that merely storing relevant information in a map is insufficient and calls for a specialized software framework to interpret map data consistently and provide different perspectives on the map. The paper aims to address these issues by presenting the Lanelet2 framework and its capabilities, focusing on the concrete structure of Lanelet2 maps and their use in the context of highly automated driving.

2. Challenges in Existing Map Formats:

The section discusses the limitations of existing map formats and frameworks used in the context of automated driving, highlighting their deficiencies in meeting the increasing demands for highly accurate, lane-level maps necessary for highly automated driving (HAD). It critiques the approaches of major commercial providers like TomTom, Here, and OpenStreetMap (OSM), explaining how these formats use an imaginary center line representation and add attributes to it, which leads to implicit and complex representations.

The paper discusses shortcomings in available formats like OpenDRIVE, pointing out their lack of freely available libraries and difficulties in data interpretation. It also examines map representations introduced for specific applications, such as the Unified Map for trajectory planning, but notes that these formats are limited in their application to certain scenarios.

Furthermore, the section mentions the dearth of suitable map formats for localization, emphasizing the need for detailed information about observable landmarks which current formats lack. It briefly discusses the map framework Liblanelet, highlighting its strengths in motion planning but also its limitations in supporting various applications and specific maneuvers.

The paper performs an analysis of 32 publications in the field of highly automated driving over the last five years, which explicitly use high-resolution maps. The majority of these publications dealt with localization and used self-created map formats or extended existing ones, suggesting a lack of suitable standardized formats. Some utilized OSM, Liblanelet, OpenDRIVE, or maps from HERE, indicating a diversity of choices and the absence of a widely accepted standard.

This analysis underscores the importance of maps in various applications related to automated driving. However, it also reveals the prevalent trend of creating or adapting map formats, indicating a deficiency in available standardized solutions. The paper suggests the need for a common, public map framework to address the challenges in highly automated driving, and it proposes to address these requirements using Lanelet2.

3. Requirements for Highly Automated Driving Maps:

This section elaborates on the essential requirements for maps in the context of highly automated driving (HAD) and discusses proposed solutions by examining common applications that rely on maps in automated driving. It classifies these applications into three categories: road network, lane and environment, and physical elements.

A. Road Network:
1. **Routing:** Emphasizes the need for greater accuracy, specifying not only which roads form a route but also the specific lanes within the roads, lane change possibilities, and alternative routes.
2. **Behavior Generation:** Requires a precise knowledge of traffic rules and lane information to generate consistent behavior following regulations.
3. **Prediction of Other Road Users:** Similar to behavior generation but with the consideration of maneuvers specific to different road users, like buses, pedestrians, cyclists, and trams.

B. Lane and Environment:
1. **Path Planning:** Stresses the importance of detailed lane geometry for adaptive trajectory planning, especially in urban environments with obstacles.
2. **Special Maneuvers:** Highlights the need for detailed map information for special maneuvers like parking or evasive actions in emergency situations.

C. Physical Elements:
1. **Localization:** Emphasizes the necessity of observable elements in the map for precise localization using various sensor setups and sufficient density of elements for accurate positioning.
2. **Data Updatability:** Addresses the challenge of keeping maps updated in the face of changes in the environment, focusing on maintaining the lineage of map information and its sources for accurate interpretation.

D. Consequences for Maps:
1. **Physical Layer:** Refers to real observable elements like markings, curbstones, and traffic lights.
2. **Relational Layer:** Abstract associations between physical layer elements, used for tracing information on the map.

The section underlines the advantages of having a clear distinction between the physical and relational layers of the map, ensuring transparency in traffic rules, accurate localization, and reliable simulation of highly automated vehicles based on the map data.

5. Lanelet2 Framework Overview:
The Lanelet2 map framework, developed with the considerations outlined in Section III, represents an extension and generalization of the Liblanelet map format. It divides the map into three layers: the physical layer (real observable elements), the relational layer (associations between physical layer elements), and the topological layer, inferred from the relational layer's contexts and relationships.

This framework, designed to be represented on an XML-based OSM data format, ensures the data format's interchangeability without loss when transferred to the internal representation. It assumes a predominantly flat ground plane projection for map elements but also accommodates height information for accurate representation.

6. Lanelet2 Framework Components and Modules:
A. Lanelet Primitives:
1. Points: Represent three-dimensional positions such as poles or other vertical structures.
2. Linestrings: Ordered arrays of points used to define the shape of elements in the map, like road markings, curbs, or fences.
3. Lanelets: Atomic sections of directed motion, defined by left and right borders, expressing traffic rules specific to that section. Lanelets can overlap or intersect.
4. Areas: Sections of the map where undirected or no movement is possible, like parking areas or green spaces. They have closed outer borders and possible inner borders.

5. Regulatory Elements: Define traffic rules, referencing lanelets or areas to which they apply, such as speed limits, priority rules, or traffic lights.

The Lanelet2 map framework comprises several essential modules, each serving specific functions:

1. Traffic Rules Module: This module interprets the rules contained in the map based on the type of road user and country. It determines the feasibility of lane changes and specifies permissions for specific road users to enter a lanelet.

2. Physical Module: This module facilitates direct and convenient access to the elements of the physical layer within the map. It likely provides functionalities to interact with and retrieve physical map elements.

3. Routing Module: Utilizes the information derived from traffic rules to create routing graphs. It aids in determining the precise route to be driven, including possible lane changes, predicting routes and potential conflict points for other road users. This module can also generate maneuverable zones by combining adjacent areas and lanelets.

4. Matching Module: Designed to assign lanelets to road users or determine potential positions on the map based on specific observations collected from the sensors. It may include algorithms or methods to correlate sensor data with the map elements.

5. Projection Module: Houses functionalities for converting global latitude/longitude coordinates to local metric coordinates. This conversion is crucial for the seamless integration of external geographic data into the Lanelet2 framework.

6. Input/Output (IO) Module: Contains functionalities for reading and writing maps from various map formats, particularly the OSM format. This module enables Lanelet2 to interface with different file structures, facilitating data import and export.

These modules together enable Lanelet2 to manage and interpret map data effectively, providing diverse functionalities that support various aspects of map utilization for highly automated driving scenarios.

2.Explore potential prediction or classification models to determine at which timestamps of total simulation time and lanes/positions will collision happen under specific configurations

When working with SUMO output data to predict collisions or accidents, several models can be suitable depending on the characteristics of the data and the specific prediction task. Here are five models that can be effective when used with SUMO output data:

Long Short-Term Memory (LSTM) Networks:

Usefulness: LSTMs are beneficial for handling sequences of data, making them suitable for SUMO output, which often involves time-stamped information. They can learn dependencies in the data over time, capturing intricate patterns and relationships between vehicle positions, velocities, accelerations, and environmental factors. LSTMs can predict collision occurrences at specific timestamps by considering the historical sequence of events leading up to the collision.

Random Forest:

Usefulness: Random Forest models can handle complex interactions in the data. In SUMO output, various parameters such as vehicle positions, speeds, distances between vehicles, environmental conditions, and traffic light states can interact in non-linear ways. Random Forest models can effectively capture these interactions, making predictions based on multiple decision trees.

Support Vector Machines (SVM):

Usefulness: SVMs are good for handling high-dimensional data and finding an optimal hyperplane for classification tasks. In SUMO output data, features like vehicle positions, accelerations, and environmental conditions can be transformed into a higher-dimensional space, allowing SVMs to find boundaries between collision and non-collision instances.

XGBoost (Gradient Boosting):

Usefulness: XGBoost is a powerful ensemble method that combines multiple decision trees to enhance predictive accuracy. In SUMO output, it can effectively handle feature interactions and capture patterns in data. It's particularly useful for scenarios where there might be complex relationships between various factors leading to collisions.

Recurrent Neural Networks (RNN):

Usefulness: RNNs, like LSTMs, are designed to handle sequential data. In SUMO simulations, RNNs can predict collision occurrences based on the historical sequence of events leading to the collision. They can analyze the time series nature of the data to predict when and where collisions might happen based on the temporal dependencies in the data.

Logistic Regression:

I utilized SUMO output data to create a basic classification model employing logistic regression. The model incorporates various features extracted from the simulation, including details such as vehicle coordinates, speed, driving behavior types, lane positioning, and indicators of collision occurrences.

Features:

x, y: These likely represent the coordinates of the vehicle in the simulation, indicating its position on a 2D plane (e.g., a map or road network).

angle: Indicates the orientation or direction the vehicle is facing. This could be an angle in degrees.

type: Describes the driving behavior or type of the vehicle, such as "RecklessDriver," "AggressiveLanechangingdriver," "HumanLikeDriver," or "CautiousDriver."

speed: Represents the speed of the vehicle, potentially in a unit like kilometers per hour (km/h) or meters per second (m/s).

pos: Possibly the position or distance of the vehicle from a reference point or location.

lane: Indicates the lane in which the vehicle is positioned. It could be represented as a lane identifier or a categorical variable defining the lane.

slope: May refer to the inclination or angle of the road on which the vehicle is driving.

acceleration: The rate of change of the vehicle's speed over time, indicating how quickly the vehicle is speeding up or slowing down.

collisions: A binary (0/1) variable indicating whether a collision occurred (1) or not (0).

victim_id: Possibly an identifier for the vehicle involved in a collision if the 'collisions' variable is marked as 1.

victim_speed: The speed of the vehicle identified as a victim in a collision scenario.

nearest_vehicle_id: Identifier of the nearest vehicle to the current vehicle in the simulation.

least_distance: Represents the smallest distance between the current vehicle and the nearest_vehicle_id.

Analysis:
The model shows exceptional performance in terms of precision and recall for class 0 (no event occurrences), correctly identifying all instances with high precision and recall.
For class 1 (event occurrences), while the precision is perfect (1.00), the model has a lower recall (0.80), indicating that it's slightly less successful in identifying all actual positive instances.
The F1-score, a balance between precision and recall, is also high, showing an overall good performance, especially for class 0. However, for class 1, while precision is perfect, the lower recall impacts the overall score.
Overall, the model demonstrates very high accuracy in detecting the absence of events (class 0), but it might need improvements to better capture and recall instances of the positive class (class 1). Adjustments could focus on improving recall without sacrificing precision, especially for the instances related to events (class 1).